IoT e MQTT: Um estudo a respeito da segurança das aplicações que utilizam o protocolo

João Pedro Brandão e Silva[1]

[1]Grupo de Resposta a Incidentes de Segurança - UFRJ, Rio de Janeiro, RJ, Brasil

Atualmente o enorme e crescente número de aplicações de Internet das Coisas têm despertado interesse em especialistas em Segurança da Informação. Desde o ataque com a Botnet Mirai a preocupação com ataques vindo de dispositivos de IoT deixou de ser vista como algo futurista para ser vista como um problema da atualidade. Com isso, a partir do grande número de dispositivos IoT que utilizam o protocolo MQTT para estabelecer a comunicação, é interessante estudar o mecanismo de funcionamento do protocolo bem como suas vulnerabilidades para que possa ser feita a devida proteção em caso de ataques futuros à aplicações de IoT.

MQTT | IoT | Internet das Coisas | Segurança da Informação |

Correspondence: ***jp.brs@poli.ufrj.br***

Introdução. O protocolo MQTT é um protocolo da camada de aplicação que, assim como HTTP, funciona em cima da pilha TCP/IP; Ele foi criado no final da década de 90 pela IBM e tinha por objetivo vincular sensores em pipelines de petróleo a satélites. Como o protocolo foi criado justamente para operar em sensores, ele foi criado com um pensamento de que iria ser usado em hardwares muito limitados e com baixa capacidade de processamento, sendo necessário que ele fosse um protocolo leve e com consumo de energia relativamente baixo.

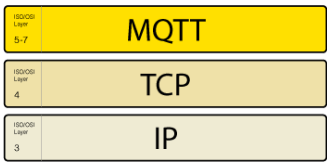

Fig. 1. NanoJ framework. Currently NanoJ consists of 5 modules dedicated to super-resolution imaging and analysis.

Mecanismo do MQTT. A comunicação de dispositivos através do protocolo MQTT se dá através de um MQTT Broker, que é uma espécie de servidor, e ocorre com dois tipos principais de mensagens : Publish e Subscribe. A mensagem do tipo Publish é a que faz a publicação de algum conteúdo em um dos tópicos disponíveis do Broker, enquanto a mensagem de Subscribe é a que assina um determinado tópico de um MQTT Broker para que o dispositivo que assinou o tópico seja atualizado das informações que serão publicadas nele. A figura a seguir mostra o mecanismo de Publish/Subscribe

Fig. 2. NanoJ framework. Currently NanoJ consists of 5 modules dedicated to super-resolution imaging and analysis.

do MQTT no tópico "/sensor", utilizando como broker o Mosquitto, um dos brokers mais populares atualmente.

Etapas da comunicação. Para que seja possível haver comunicação entre o Broker e seus clientes, antes de fazer uma publicação ou assinatura em um tópico do broker é necessário que o cliente crie uma conexão com o mesmo. Essa conexão se dá através do pacote connect, justamente o pacote que deverá ter o maior foco nesse artigo por ser o responsável pela parte de conexão, autenticação, tempo de conexão, etc. como será demonstrado em outra seção. Ao tentar criar uma conexão com o Broker, em resposta ao pacote Connect enviado pelo cliente, haverá um pacote Connack enviado pelo Broker, que possuirá em seu conteúdo o estado da conexão e um código de retorno. O cliente pode receber seis tipos de códigos de retornos diferentes, cada um com a seguinte descrição :

Table 1. Connection Codes

Code	Description
0	Conexão Aceita
1	Conexão Recusada - Versão do protocolo não aceita
2	Conexão Recusada - Identificador Rejeitado
3	Conexão Recusada - Servidor Indisponível
4	Conexão Recusada - Usuário ou senha inválidos
5	Conexão Recusada - Não autorizado

Após receber o pacote Connack, com código de retorno Zero, a conexão entre cliente e broker é estabelecida e portanto as publicações e inscrições podem ser feitas. Atentando-se para o detalhe de que apenas as publicações feitas após o momento de inscrição poderão ser consumidas pelo cliente que assina o tópico. Após o uso dos recursos do Broker há sempre a opção de se desconectar intencionalmente e encerrar a comu-

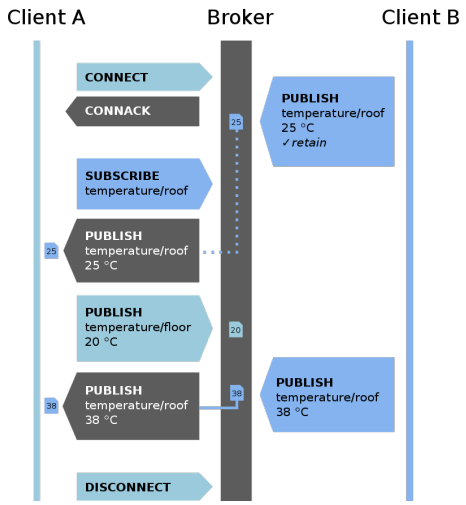

Fig. 3. Exemplo de uma comunicação Publish-Subscribe

nicação, caso isso não seja feito a comunicação se fechará de um jeito ou de outro após um tempo determinado. Um exemplo de comunicação entre dois clientes através de um Broker será mostrado a seguir na Figura 3.

Pacote Connect e Configurações de Broker. Como já foi dito, o pacote connect é o pacote enviado que é responsável não só pela conexão, mas também pela autenticação dos clientes no MQTT Broker. Os principais campos do pacote connect que serão discutidos nesse artigo serão "clientId", "username", "password" e "keepAlive", justamente por serem os pacotes mais relacionados à segurança. Um exemplo de pacote connect pode ser visto na figura 4.

A. clientId. É o campo que permite ao Broker identificar o Cliente e relacioná-lo com o estado atual da conexão deste. A existência de um clientId pode ser opcional(vazio) ou não, dependendo das configurações do Broker. O ClientID vazio resulta em uma conexão sem nenhum estado, o que só torna possível a aceitação da conexão em casos que o sinalizador cleanSession esteja configurado como true. Porém, por questões de segurança deve-se desabilitar a possibilidade de haver clientIds anônimos nas conexões MQTT, e isso deve ser configurado no próprio MQTT Broker. No caso do Mosquitto Broker isso pode ser feito adicionando a linha "allow_anonymous false" no arquivo "mosquitto.conf" que em sistemas operacionais linux geralmente se encontra em "/etc/mosquitto/". Isso faz com que cada cliente precise de um clientId único, pois torna inviável a existência de mais de uma conexão feita pelo mesmo clientId. Uma última observação sobre o campo é que pode-se configurar o broker para permitir que somente um determinado clientId possa assinar e/ou publicar em determinado tópico.

B. username e password. Esses dois campos também são opicionais, porém por motivos óbvios é mais que recomendado que o MQTT Broker esteja configurado apenas para receber conexões com determinados usuários e suas respectivas senhas. Esse tipo de configuração pode ser feito no Mosquitto Broker da seguinte maneira com as linhas de código a seguir. Primeiro deve-se usar o comando:
$mosquitto_passwd -c /etc/mosquitto/passwd <user_name>
Com isso será pedido justamente a senha que se gostaria de associar ao usuário, após isso deve-se digitar novamente a senha para que a configuração seja efetivada.Por último deve-se adicionar o arquivo criado com as Hashes das senhas relacionadas aos usuários criados ao arquivo de configurações do MQTT. Isso pode ser feito facilmente com o comando abaixo:
$echo "password_file /etc/mosquitto/passwd" » /etc/mosquitto/mosquitto.conf
Outro ponto interessante sobre o campo de usuário é que pode-se configurar o broker para permitir que somente um determinado usuário possa assinar e/ou publicar em determinado tópico, assim como pode ser feito com o clientId.

C. keepAlive. O campo keepAlive é o campo que contém o tempo em que uma conexão MQTT ficaria aberta dado que não há fluxos de dados durante esse meio tempo; possuindo como tempo máximo 18:12:15. Como o MQTT é baseado no TCP, a necessidade de um keepAlive foi crescendo visto que o problema de conexões semi-abertas do TCP aumenta consideravelmente quando estamos falando de rede movéis. Como o MQTT foi desenvolvido justamente para se comunicar com satélites, consequentemente os problemas de conexões semi-abertas iriam acabar surgindo com uma certa frequência.

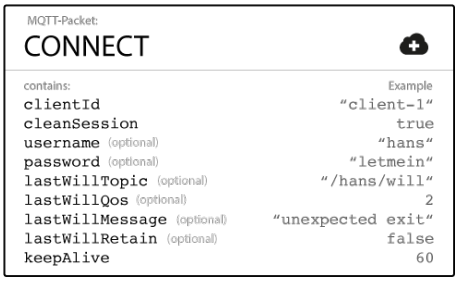

Fig. 4. Estrutura de um pacote Connect

Um dos problemas das mensagens connect do MQTT é que por padrão o cabeçalho e os dados são enviados sem nenhum tipo de criptografia, isto é, o MQTT quando funcionando na porta padrão, que é 1883, tem seus pacotes vulneráveis e consequentemente com isso a confiabilidade do mecanismo de autenticação diminui. Vale ressaltar o fato que numa rede de dispositivos IoT o número de dispositivos acaba sendo relativamente grande em comparação às outras aplicações, o que justamente implica que muitos dispositivos virão a enviar esses pacotes connect em algum momento, tornando mais fácil a interceptação

MQTT e TLS. Além de existir a tradicional maneira de encriptar os payloads dos pacotes para que somente o dispositivo final consiga ter acesso a informação, o MQTT também oferece nativamente a possibilidade de se usar criptografia TLS em sua comunicação, mudando assim a porta de comunicação para 8883. A criptografia pode ser usada dando

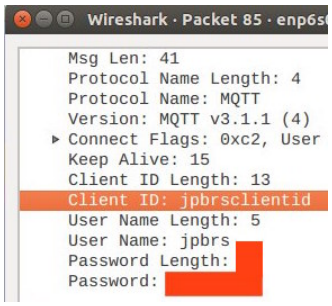

Fig. 5. Pacote Connect interceptado com Wireshark

ao MQTT Broker um certificado vindo de uma autoridade certificadora, além das devidas configurações por parte do cliente. Existem alguns problemas em relação ao uso do TLS, o primeiro deles é justamente a necessidade de ser reconhecido por autoridade certificadora, visto que elas só distribuem tal certificado para Brokers que possuem domínio público. Isso pode ser contornado por exemplo utilizando um DNS dinâmico, como Duck DNS, atualizando sempre que houver mudança no IP relacionado ao domínio. Dessa forma pode-se obter o certificado até mesmo de forma gratuita, utilizando o Let's Encrypt por exemplo. O segundo e maior problema relacionado ao TLS em dispositivos IoT é justamente a necessidade de processamento extra em que os dispositivos são submetidos ao utilizar tal tipo de comunicação. A maior parte das aplicações IoT prezam pelo tempo de vida útil do dispositivo, isto é o tempo que uma bateria dura alimentando os dispositivos de tal aplicação. O TLS faz com que o processamento e consequentemente o consumo de energia aumente muito, além do que por exigir uma capacidade de processamento relativamente alta para dispositivos IoT, não são todos os dispositivos que conseguem utilizar o protocolo MQTT sobre o TLS. Na figura 5 pode-se ver um scan feito no dia 27/06/2019 utilizando o Shodan com a query "port:1883 "MQTT"". Como pode ser visto, havia 79.000 MQTT Brokers conectados a internet e que não utilizavam TLS seja pelos motivos acima citados ou simplesmente por desconhecimento ou negligência.

Fig. 6.

Outra coisa interessante que foi revelada pelo Scan é que boa parte dos MQTT Brokers revelados retornaram connection code 0, ou seja, permitiram o Shodan a se conectar com o Broker sem nenhum tipo de autenticação. Isto é preocupante pois mesmo que não se saiba o endereço físico ou o viés da aplicação, existe a possibilidade de se ver todas as mensagens publicadas nos tópicos que não sejam de configuração(que começam com $) apenas com uma simples inscrição no tópico '#'. Em relação aos tópicos de configuração, também há a possibilidade de se obter informações sobre o Broker e sobre a aplicação com mensagens de Subscribe sobre eles. Assinando por exemplo o tópico do mosquitto "$SYS/broker/version" pode-se ver a versão do broker, enquanto "$SYS/broker/clients/connected" pode-se ver o número de clientes conectados ao broker. Além desses existem diversos tópicos que retornam informações importantes sobre o Mosquitto broker que podem ser exploradas em ataques e estão expostas em boa parte dos Brokers MQTT conectados à internet.

Home Assistant. Boa parte de Brokers expostos irresponsavelmente à internet são brokers responsáveis pela automação de casas, que funcionam com, por exemplo, o software Home Assistant. A automação do Home Assistant funciona de maneira bem simples e por isso o software vêm se popularizando inclusive com pessoas que não possuem muito conhecimento na área. Numa aplicação de automação com Home Assistant existem microcontroladores que são responsáveis por ligar ou desligar algum dispositivo; esses microcontroladores assinam um determinado tópico do Broker esperando que seja publicada uma mensagem de gatilho como por exemplo "On" ou "Off" para executar a instrução de ligar ou desligar um dispositivo. Ou seja, visto que pode-se fazer a inscrição em todos os tópicos com Subcribe "#", pode-se ver que tipo de mensagem é publicado em cada tópico e consequentemente ter controle de aplicações IoT de automação que funcionam com esse mecanismo e que não possuem segurança no MQTT Broker assim como as que foram reveladas pelo Shodan. Outro ponto interessante sobre o Home Assistant é que sua imagem permite a instalação de outros softwares, como por exemplo o popular OwnTracks, que atualiza o HA com as coordenadas do celular cadastrado. Isso pode ser usado para, por exemplo, regular a temperatura da casa quando for visto que o usuário está perto de chegar, para que quando ele finalmente chegasse, o ambiente estaria climatizado. Então, utilizando o mesmo mecanismo de Subscribe, é possível assinar o tópico que ocorre as publicações das coordenadas do celular do usuário para o uso do OwnTracks e assim ter acesso à geolocalização do usuário sem nenhuma dificuldade, visto que o payload das publicações é em formato json e possui os atributos "latitude" e "longitude".

C.1. DoS. O ataque de negação de serviço sobre o Mosquitto Broker pode ser feito a partir da abertura de diversas conexões MQTT entre um cliente que utiliza client_id's diferentes para cada conexão e um broker que mantêm essas conexões vivas até que o KeepAlive determine que aconteça a desconexão. O Mosquitto Broker trata cada conexão feita por um cliente como uma Thread no Sistema Operacional Linux, porém não é possível fazer com que o Mosquitto Broker crie um número ilimitado de Threads para atender à todas as conexões possíveis. Sendo assim, lógicamente existe um número máximo de Threads ou conexões que podem

ser abertar por um Mosquitto Broker, testando num RaspBerry 3b+ pode-se ver que o número de threads que podem ser criadas é aproximadamente 1250(um número maior que por exemplo o Apache, que em mesmas condições cria 400 conexões). Sendo assim, um ataque vindo de apenas uma máquina com acesso ao Broker pode facilmente criar uma negação de serviço da aplicação IoT, basta que dada a condição mais básica tenha pelo menos como pré-requisito de cada conexão um client_id único, para que o Broker crie uma nova Thread a cada nova conexão.
Tentar utilizar o Tor sorbe MQTT e utilizar Verbatim

C.2. Botnet de SONOFF. [1]

Como foi visto na seção anterior, os microcontroladores de uma aplicação de automação residencial controlar os dispositivos a partir de mensagens de Subscribe em um tópico do broker. Porém, muitos microcontroladores, como por exemplo os ESP8266 encontrados em tomadas inteligentes como Sonoffs, que são muito populares, além de assinar o tópico designado para controlar os dispositivos, tabém assinam outros tópicos do Broker que são responsáveis pela configuração do próprio Sonoff. Por exemplo, existem os tópicos responsáveis pela configuração de wifi do Sonoff como "cmnd/sonoff/Ssid" e "cmnd/sonoff/Password" e existe um tópico em especial chamado "cmnd/sonoff/otaUrl". OTA, ou Over The Air, é uma opção que os microcontroladores ESP8266 possuem para que seja possível escrever em seu firmware sem que seja preciso usar um fio conectando o computador ao microcontrolador. Isso é muito útil visto que esses dispositivos nem sempre estão em lugares acessíveis e portanto torna-se muito mais prático programá-lo com a opção de OTA ativada. O problema disso no caso dos Sonoff é justamente que é possível publicar uma Url nesse tópico para que o Sonoff baixe um código por essa Url. Sendo assim, publicado uma Url maliciosa é possível escrever no firmware do dispositivo e programá-lo para, por exemplo, esperar uma publicação em um tópico "mqtt/ddos" cujo payload seria um endereço IP, fazendo com que o dispositivo, que já está com o código malicioso dentro dele, comece a fazer um ataque de negação de serviço no servidor especificado.

Supplementary Note 1: References

1. HiveMQ website. *HiveMQ*. https://www.hivemq.com/
2. Syaiful Andy, Budi Rahardjo, Bagus Hanindhito. *Attack Scenarios and Security Analysis of MQTT Communication Protocol in IoT System*
3. GTA-UFRJ website. *GTA UFRJ*. https://www.gta.ufrj.br/ensino/eel878/redes1-2018-1/trabalhos-vf/mqtt/
4. IBM website. *IBM*. https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html
5. Moshe Zioni. *"Don't let the cuteness fool you: Exploiting IoT's MQTT protocol", AppSecIL 2017.*
6. Giovanni Perrone , Massimo Vecchio , Riccardo Pecori and Raffaele Giaffreda. *The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices*
7. Dipa Soni, Ashwin Makwana *A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)*